A warning about the NDDL string type

In NDDL, the *string* data-type represents the infinite domain of all possible strings. In the implementation, this does not behave the way most users want and expect it to, because the default domain is empty but open, and string domains aren't "linked" in the same way as open object domains are (adding a value to one string domain won't cause it to be added to all existing and future string domains). NDDL doesn't provide a way to insert values into domains, so a a string must be initialized (set a member variable in a constructor, for example), and cannot easily be reset later (because the domain will only include that initial value).

Strings can therefore be appropriate for id-type member variables that will be set once on construction and never changed. If you instead want to be able to update/change the value of a *string* variable, you can:

- Use symbolic enums instead, for cases where the values of the domain are known.
- Similarly, you could specify possible values with each variable instance (although this is just like an enum with the disadvantage that you can't specify the domain once):

  ```
  string x = {"lions","tigers", "bears"};
  ```
- Write custom code to update the domains when necessary.
- Use the StringData class or a sub-class of it to get the base-domain updating behavior of open object types.

NOTE: We plan to update *string* to behave as expected, but this is a significant undertaking, and we don't want to make promises about when it will be done.